those rules on the data.

In one embodiment of the present invention, data models are expressed using structured markup language notation. As one example, XML may be used to encode data model objects which include data values as well as data validation. (It should be noted that references herein to XML are for purposes of illustration and not of limitation.) Fig. 3 illustrates a sample validation object 300 for use when storing a social security number, and the rules contained therein are designed specifically for social security numbers. In particular, the rules reflect that such numbers contain exactly nine digits, and may optionally contain dashes, but are not allowed to contain any other types of special characters or letters. Validation object 300 will now be described in more detail, as an example of the general principles of the present invention.

According to preferred embodiments of the present invention, validation rules for string data specify a minimum length and a maximum length as attributes; in the example of Fig. 3, the length restriction is specified using the value of "9" for both the minimum and maximum (see elements 315 and 320). The tag "stringVariable" indicates that the rules 305 pertain to a string data type. In addition to minimum and maximum length attributes, each string tag includes a name attribute 310. Optionally, string tags may also include attributes specifying string-related information such as whether data values are to be enforced as all upper case or all lower case; by default, mixed case is preferably allowed.

Optionally, the string tag may include a label child tag 325 which specifies how the

associated data should be described on a widget (such as a button or entry field) that may be generated on a GUI. In the example, a label value "SSN" has been provided. Additional optional child tags are a help text tag 330, which provides help text (or a reference to stored help text, using for example a Uniform Resource Locator or other file identifier) pertaining to the associated data, and a mnemonic tag 335, which defines a hot key value to be used for this widget on the GUI.

The input validation rules for the social security number example specified in the "inputValidation" tag 340 include ignoring the dash characters (see 345) and allowing as valid characters only entry of digits and dashes (see 350).

The validation object 300 may be associated with any variable in which a social security number is to be stored. It will be obvious how this approach may be used to specify validation rules for other types of string variables as well as for other data types. Preferably, other data types use attributes and child tags which are adapted to those data types. Additional or different data types may be supported without deviating from the inventive concepts disclosed herein.

According to preferred embodiments of the present invention, the validation object for each data type includes a name attribute which names the variable to which this validation object applies, and the label text, help text, mnemonic, and inputValidation child tags preferably apply to each data type as described with reference to Fig. 3.

The validation may be activated by various events, as reflected by an application developer. For example, validation may be invoked when a user clicks on a button, or when a window or widget loses focus (indicating that the user has moved on to another window or widget), and so forth.

Turning now to Fig. 4, logic underlying an implementation of the present invention is provided. As shown in Block 400, a GUI window is opened for user interaction. A data model is associated with a widget from this GUI (Block 405), and the user then interacts with that data model through the GUI widget. When the window closes (Block 410), all the window's widgets preferably invoke their validation methods (Block 415). (As discussed earlier, validation may alternatively be triggered at other times or in response to other events.) In Block 420, the widget then delegates the validation to the data model, according to the present invention. The custom validation defined in the data model is applied (Block 425), and the results are returned to the widget (Block 430). The widget then preferably returns these results to the window (Block 435). The processing of the current invocation of Fig. 4 then ends.

Sample code which may be used to perform validation of data according to preferred embodiments of the present invention is shown in Figs. 5A and 5B. As can be seen from inspection of this code, the data model recognizes that data has been changed (for example, when a user provides a revised data value) through a variable changed event. When this event is triggered, it launches the validation process. The data model has standard local validation in a "doValidate" method. It then looks for installed instances of a custom validator, such as those